AGILE SOFTWARE DEVELOPMENT PRACTICES THAT INFLUENCE SOFTWARE QUALITY: A REVIEW

1Shafinah Farvin Packeer Mohamed, 2Fauziah Baharom, 3Aziz Deraman, 4Jamaiah Yahya

Universiti Utara Malaysia, 1shafinah@uum.edu.my, 2fauziah@uum.edu.my, 4jamaiah@uum.edu.my
Universiti Malaysia Terengganu, 3nc@umt.edu.my

ABSTRACT. Agile software development (Agile) is being practiced in software industry nowadays as it fits the current business environment which focuses on delivering software to market as quickly as possible. In addition, Agile practitioners claim that it produces software with good quality. Thus, our research aims to identify Agile practices that should be followed in order to produce good quality software. Since many researchers report that the quality of people and process influence the quality of software product, this paper discusses on practices related to these two factors. The identified practices will be used for developing questionnaire in order to investigate current practice among Agile practitioners.

Keywords: Agile Software Development, Software Quality, Software Practice.

INTRODUCTION

Software quality has become a major strategic issue in software industry (Jamaiah, Fauziah, Aziz & Abdul Razak, 2005). This is because customers always expect that the software product, service and process to be good in quality (Lycett, Macredie, Patel & Paul, 2003), which meets their needs and follows certain standards (Krishnan, 1993). Besides, they also expect that software products can be developed faster (Verner, Liming, Babar & Ming, 2004). Generally, software with good quality has these criteria: 1) meets the expected requirements, 2) completed within budget, 3) completed on time, 4) completed in its entirety, 5) delivered together with a solid and thoroughly tested code, and 6) can be used easily (Nasution & Weistroffer, 2009). The quality of a software product highly depends on the people, organization and procedures used to create and deliver it (Fuggetta, 2000). However, according to Arthur (1993) and O’Regan (2002), there are three key elements need to be given attention in developing good software: 1) the quality of people involved, 2) the process performed and 3) the use of development technology. Nevertheless, Fauziah (2008) identified another two factors that influence software quality which are the working environment, and project condition. Besides, Hazzan and Dubinsky (2009) stated that human, technology used and organizational aspects should be considered in assuring software quality. Based on the literature mentioned, factors that influence software quality can be classified into five, which are process, human, working environment, technology and project condition.

Realizing the needs for faster software development cycle and rapidly changing requirements, many organizations are shifting from conventional software development approach to Agile which is considered as a light-weight approach (Conn, 2004). However, most Agile opponents argue that this software development approach is lack of
documentation, which leads to software maintainability problem. In addition, Turk, France and Rumpe (2002) mentioned some limitations in Agile, such as, limited support for distributed development environments and limited support for building reusable artifacts. They believe these limitations able to give impact on the quality of developed software. On the other hand, many Agile representatives claim that it fits the industrial needs (Beck, 2000) because it promises higher quality software (Sliger, 2006), higher customer satisfaction, lower defect rates, faster development times and becomes a solution to rapidly changing requirements (Boehm & Turner, 2003). Consequently, this paper reviews the literature related to practices of Agile development approach that must be performed in producing good quality software as part of our research work. Our research aims to construct a unified software process certification model which can be used to assess and certify software based the quality of development process no matter what approach was used either conventional, Agile or Web. However, discussion in this paper will focus on findings from literature about the required practices related to human and process factors of Agile based development. The structure of this paper is organized as follows: the next section discusses about Agile practices that influence software quality and finally future work and conclusion is provided.

AGILE SOFTWARE DEVELOPMENT PRACTICES THAT INFLUENCE SOFTWARE QUALITY

Agile is introduced recently as a consequence from the problems faced in conventional methodologies (Rico, Sayani & Sone, 2009) which are not flexible in accepting unstable and volatile requirements (Verner et al., 2004). It is aimed to produce higher quality software in a shorter period of time (Livermore, 2007; Sliger, 2006). Currently there are many Agile methodologies such as Extreme Programming (XP) and Scrum (Abrahamsson, Salo, Ronkainen & Warsta, 2002). These methodologies have similar values and practices, whereby they follow 12 principles, for instance: “Welcome changing requirements, even late in development” and “Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale” (Agile Manifesto, 2001). It also follows four values which are: 1) iterative, 2) incremental, 3) self-organizing and 4) emergent (Lindvall et al., 2002).

A survey conducted by Microsoft Researchers reveals the benefits of Agile as improved communication and flexibility, with faster release. Another survey conducted by IBM and Ambyssoft shows that Agile adoption improves productivity, customer satisfaction and project success (Rico et al., 2009). The results of these studies show that Agile leads in producing good quality software. Therefore, there is a need to identify the Agile practices that influence software quality. Although factors that influence software quality can be classified into five, however as mention in previous section, this paper will only discuss two factors which are human and process.

Human

Agile implementation highly depends on human factor (Mnkandla, 2004; Lycett et al., 2003; Cockburn & Highsmith, 2001). Most literature highlight about team, project manager, developer and customer when discussing about human involved in Agile. Each of them needs to implement certain practices in sequence to ensure software quality. They are encouraged to be placed in a single space to facilitate intensive communication, because communication is emphasized for knowledge sharing (Pressman, 2005; Lycett et al., 2003; Highsmith & Cockburn, 2001). Besides, face to face communication is accepted as more effective to transfer idea compared to writing and reading documents (Highsmith & Cockburn, 2001). Smaller size team will encourage better communication, as more team members will cause the project to be ‘less agile’ (Abrahamsson et al., 2002). On top of that, most Agile methodologies does not work for large development teams. Thus, the suggested average
number of team member is nine (Highsmith & Cockburn, 2001). The practices that must be
followed are listed thoroughly in Table 1.

Process

Software development process is important in producing good quality software, as stated
by Deming (1982) “the quality of product is influenced by the quality of process used to
develop it”. Besides the development process, project management also plays an important
role. Basically Agile has all fundamental software development phases, which are
requirement gathering, design, coding and testing. However, instead of having the four phases
one after another throughout development, Agile has these phases iteratively in shorter time
(Beck, 1999). Besides that, the implementation is also different. Requirement gathering and
design in Agile is done iteratively and incrementally rather than gathering all requirements
and designing upfront. During the development process, Agile emphasizes practices which
can promote quality and faster delivery, such as pair programming and test driven
development. These practices are further listed in Table 1.

Table 3. Agile Practices that Influence Software Quality

<table>
<thead>
<tr>
<th>Factor</th>
<th>Sub Factor</th>
<th>Agile Practice</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>Project Manager</td>
<td>Engaged with daily activities</td>
<td>Schuh (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Responsible to ensure that news (bad or good) is spread between customer and team.</td>
<td>Schuh (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Knowledgeable in agile process</td>
<td>Tsun &amp; Dac –Buu(2008)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Has adaptive management style</td>
<td>Tsun &amp; Dac-Buu(2008)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Responsible to the overall project’s progress</td>
<td>Schuh (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Responsible to maintain relationship with customers</td>
<td>Schuh (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Acts more like a facilitator than a foreman</td>
<td>Sliger (2006); Schuh (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Responsible to build team cohesion</td>
<td>Sliger (2006)</td>
</tr>
<tr>
<td>Team</td>
<td>Team</td>
<td>High competence and expertise</td>
<td>Tsun &amp; Dac –Buu (2008)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Great motivation</td>
<td>Tsun &amp; Dac-Buu (2008)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Common focus</td>
<td>Pressman (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Communication is used as important mechanism for knowledge sharing</td>
<td>Coram &amp; Boliner (2005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Self-organized</td>
<td>Sliger &amp; Broderick (2008)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Co-located</td>
<td>Misra, Kumar &amp; Kumar (2009)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Empowered to make decisions</td>
<td>Sliger &amp; Broderick (2008); Lindval et al. (2002)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Decisions made collaboratively but speedy</td>
<td>Highsmith &amp; Cockburn (2001)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Able to give constant feedback</td>
<td>Highsmith &amp; Cockburn (2001)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mutual trust and respect exist among team members</td>
<td>Pressman (2005); Highsmith &amp; Cockburn (2001)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Able to deal with ambiguity</td>
<td>Highsmith &amp; Cockburn (2001)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intense interaction exist among team member</td>
<td>Highsmith &amp; Cockburn (2001)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average size of team is 9 people</td>
<td>Highsmith &amp; Cockburn (2001)</td>
</tr>
<tr>
<td>Process</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Do not feel afraid to be responsible to the decisions made | Schuh (2005)  
| Willing to compromise | Schuh (2005)  
| Planning | 
| Done collaboratively with team members | Ambler (2010)  
| Done continuously throughout the project at the beginning of each iterations and releases | Schuh (2005); Ambler (2005)  
| Done according to features/stories | Sliger & Broderick (2008); Schuh (2005)  
| There exists daily stand up meetings among developers | Sliger & Broderick (2008)  
| Release meeting is conducted at the beginning of project to create release plan | Sliger (2006); Schuh (2005)  
| Iteration plan is created at the beginning of each iterations | Sliger (2006); Schuh (2005)  
| User selects stories to be implemented in each iteration based on the estimates and velocity produced | Schuh (2005)  
| Tasks estimation must be made by the developer who is going to accomplish the task | Wells (2009)  
| Requirement Gathering | 
| Iterative requirement engineering | Wells (2009); Lan & Ramesh, (2008)  
| Face-to-face communication is emphasized instead of having written specification | Lan & Ramesh, (2008); Paetsch et al. (2003)  
| High level requirements are written in user stories(XP)/backlog(Scrum)/features(FDD and DSDM) for requirement gathering | Rico et al., (2009); Wells (2009); Lan & Ramesh, (2008)  
| Detailed requirements are discussed in detail at each development cycle’s start | Wells (2009); Lan & Ramesh, (2008)  
| Information needed on user story: its name, the story and developer’s estimation | Schuh (2005)  
| The story written should be in simple and understandable language | Wells (2009); Schuh (2005)  
| Prioritization to the user stories is done by user | Lan & Ramesh, (2008); Paetsch et al. (2003)  
| Requirements can be reprioritized by user | Lan & Ramesh, (2008); Paetsch et al. (2003)  
| Requirements can be added, removed or edited by users | Schuh (2005)  
| Use prototype to validate the requirements | Lan & Ramesh, (2008)  
| Design | 
| Software is designed in small chunks and integrated in ongoing manner | Highsmith & Cockburn (2001)  
| Agile Modeling is used to model high-level architecture of the system upfront | Ambler (2010)  
| Unit tests which is implemented for Test Driven Development is used as detailed design artifact | Ambler (2005)  
| Metaphor is used for determining architecture of the system | Rico et al., (2009)  
| Coding | 
| Implement collective code ownership | Ambler (2010); Rico et al., (2009); Wells (2009)  
| Implement coding standards | Rico et al., (2009); Wells(2009)  
| Implement pair programming | Rico et al., (2009); Wells (2009)  
| Implement code and database refactoring | Ambler (2005)  
| Unit tests are developed before the code is written | Ambler(2010); Wells (2009)  
| Group and implement requirements with highest priority first | Leffingwell (2007)  
| Ensure that the code produced is tested, working and integrated to system baseline | Wells (2009); Leffingwell (2007)  
| Unit tests are developed before the code is implemented | Leffingwell (2007); Schuh (2005)  
| Customer writes the user acceptance tests according to stories/features | Schuh (2005); Abrahamsson et al. (2002)  

150
Testing

<table>
<thead>
<tr>
<th>Test</th>
<th>Details</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>User acceptance tests are used for requirements validation</td>
<td>Lan &amp; Ramesh, (2008); Paetsch et al. (2003)</td>
</tr>
<tr>
<td></td>
<td>Continuous testing throughout development</td>
<td>Leffingwell (2007)</td>
</tr>
<tr>
<td></td>
<td>Conduct review meetings to validate Requirements</td>
<td>Lan &amp; Ramesh, (2008); Paetsch et al. (2003)</td>
</tr>
<tr>
<td></td>
<td>Pair programming promotes peer review</td>
<td>Rico et al. (2009)</td>
</tr>
</tbody>
</table>

FUTURE WORK AND CONCLUSION

Figure 10. The Conceptual Framework for Unified Software Process Certification Model

Figure 1 shows the proposed conceptual framework of our research. As mentioned earlier, this study aims to construct a Unified Software Process Certification Model. One of the components in the model is the standardized criteria which act as a benchmark of the model. The standardized criteria will be constructed by identifying the best practices that must be performed in producing high quality software product. This study will identify and consider the best software development practices of conventional, Agile and Web-based software development. Since quality and rapid development process are treated as important goals to be achieved in software industry, Agile has been widely adopted by software developers. This paper presents the identified Agile software development practices that are related to human and process factors. Results of the literature found the practices emphasized for human factor are team work, effective communication, decision-making skills, and ability to work quickly. While for the process factor, the literature shows that Agile supports all fundamental software development phases although the implementation is different (refer Table 1). The importance of these practices will be verified by Agile practitioners through an empirical study. Only the practices which get high consideration from them will be included as the standardized criteria for our model. Moreover, during the empirical study, additional practices that might be suggested by the practitioners will be taken into consideration.

REFERENCES


Proceedings of the 3rd International Conference on Computing and Informatics, ICOCI  
2011, 8-9 June, 2011 Bandung, Indonesia  
Paper No. 092


